

Rapport Écrit

Projet Final Informatique – Option B3

Adrian BULNES (eu4m@adrianbulnes.com)
David FUENMAYOR(david.fuenmayor@eu4m.eu)
Vicky LYSANDRA(vicky.lysandra@gmail.com)

1. Introduction

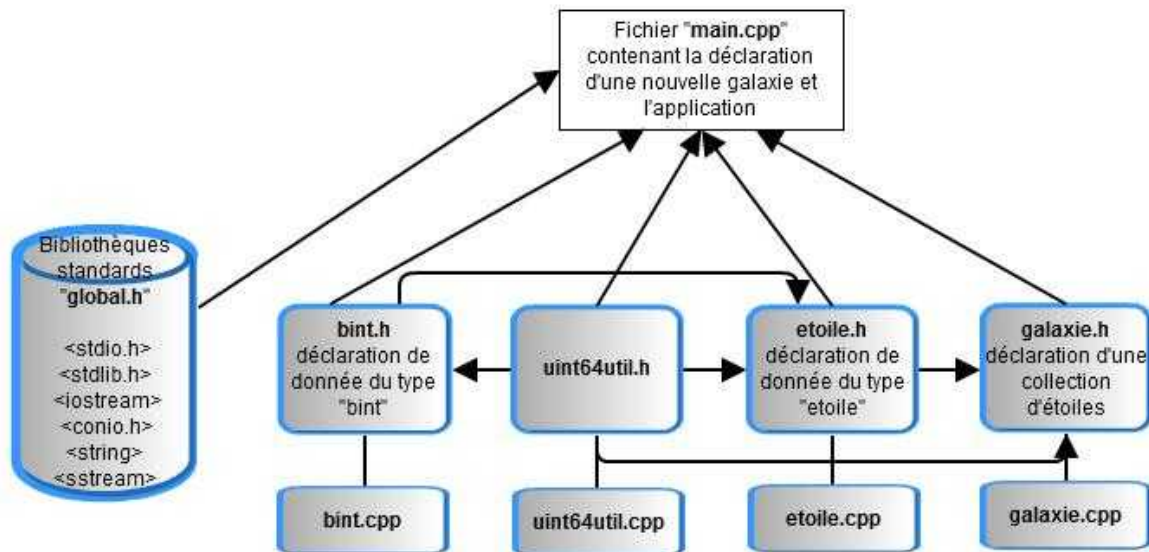
L'objectif principal du projet est de pouvoir traiter des opérations mathématiques avec nombres entiers très grands, de façon à ce qu'on puisse travailler avec une précision arbitraire, c'est à dire, ne pas perdre des chiffres dans des nombres très grandes, à cause des limitations technologiques des moyens de calcul.

Un nouveau type de donnée a été créé pour sauvegarder un entier positif de longueur variable en blocs, chacun contenant un nombre entier. Afin d'opérer sur un de ces nombres, des opérations arithmétiques sont effectués sur chacun de ses blocs composants. On se sert alors de sous-programmes développés pour conduire les opérations arithmétiques de base pour ce type de donnée.

Le fonctionnement de ce nouveau type de données est illustré avec la création d'un enregistrement appelé "étoile", dont chacun de ses composants, tels que la distance au centre de la galaxie, le rayon et le volume, est de type "bint". Ensuite, un ensemble appelé "galaxie", est implementé en tant qu'une liste chaînée d'étoiles, stockées en utilisant une liste chaînée, et triées à partir de ses distances au centre de la galaxie. Outre, il est aussi possible de modifier les étoiles membres d'une galaxie, les ajouter ou les supprimer, selon le cas souhaité, à partir d'un menu principal.

2. Développement

Le projet est composé des modules suivantes:



2.1 Fichiers *bint.h* et *bint.cpp*

Un type de donnée appelé "bint" à été créé, qui se compose d'un grand entier et les informations additionnels utiles pour son utilisation.

//Enregistrement qui represente un entier positive de longueur quelconque (longueur interne dynamique)

```

typedef struct bint {

    // blocues (array)
    unsigned long long *b;

    // longueur. Nombre de blocues utilises
    unsigned int nb;

    // longueur reel. Longueur d'array
    unsigned int internal_nb;

};
  
```

Les sous-programmes utilisés pour traiter "bint" se trouvent dans le fichier "bint.cpp". Là on compte avec les opérations suivantes:

2.1.1. Opérations de base:

// creer un bint vide avec la valeur 0

```

bint bint_creerVide();

// créer bint à partir d'un uint64
bint bint_creer(unsigned long long valeur);

// assigner un uint64 à un bint
void bint_assigner(bint &b, unsigned long long n);

// copier un bint
bint bint_copier(const bint &a);

```

2.1.2. Les fonctions d'affichage

```

// Information générale : DEBUG
void bint_info(bint &b);

// Obtient la valeur binaire
string bint_getBin(bint &b);

// Compte le nombre de bits
unsigned long long bint_compterBits(bint &b);

// Retourn une valeur approximative
string bint_valEnviron(bint &b);

// Convert bint en une chaîne
string bint_toString(bint &b);

```

2.1.3. Les procédures pour les traitements arithmétiques

```

// a=a+b
void bint_ajouter(bint &a, int b);
void bint_ajouter(bint &a, unsigned long long b);
void bint_ajouter(bint &a, const bint &b);

// ajoute un nombre en certain position du array
void bint_ajouter(bint &a, unsigned long long b, int pos);

// a=a-b
void bint_sustraire(bint &a, int b);

```

```

void bint_sustraire(bint &a, unsigned long long b);

// sustraire un nombre en certain position du array
void bint_sustraire(bint &a, unsigned long long b, int pos);

// a=a*b
void bint_multiplier(bint &a, unsigned int b);
void bint_multiplier(bint &a, const bint &b);

// a=a^b
void bint_pow(bint &base, unsigned long long exp);
void bint_pow(bint &base, const bint &exp);

```

2.1.4. Les fonctions pour faire des comparations entre valeurs

```

// est vide or ==0
bool bint_estVide(const bint &a);

// TRUE si a>b. FALSE si a<=b
bool bint_plusGrandQue(const bint &a, unsigned long long b);
bool bint_plusGrandQue(const bint &a, const bint &b);

// TRUE si a<b. FALSE si a>=b
bool bint_plusPetitQue(const bint &a, unsigned long long b);
bool bint_plusPetitQue(const bint &a, const bint &b);

// TRUE si a==b
bool bint_egal(const bint &a, unsigned long long b);
bool bint_egal(const bint &a, const bint &b);

// TRUE si a==b et les arrays sont egales aussi
bool bint_egalExact(const bint &a, const bint &b);

```

2.1.5. Changement de valeurs internes du bint

```

// options internes
// changer la longueur interieure de bint
void bint_setLongInter(bint &a, unsigned int L);

```

2.2 Fichiers *etoile.h* et *etoile.cpp*

En utilisant "bint", on a construit un autre enregistrement appelé "etoile" dans le fichier "etoile.h". Où chaque "etoile" possède l'information suivante:

- La distance au centre de la galaxie
- La taille (rayon)
- Le volume de l'etoile

Dont chaque composant est implementé en tant qu'un type "bint":

// Un type de donnée appele "Etoile", utilisee pour montrer l'utilisation du type "bint"

```
typedef struct etoile {  
    // distance au centre de la galaxie (1012 km)  
    bint distCentGal;  
    // rayon (radius) de l'etoile (103 km)  
    bint rayon;  
    // volume de l'etoile (103 km3)  
    bint volume;  
};
```

En tant que fonctions pour la création et le traitement d'une étoile, l'on compte sur:

// Cree une etoile a partir de la distance au centre du galaxie et son rayon

```
etoile etoile_crear(unsigned long long distanceCentreGalaxie,  
unsigned long long rayon);
```

// creer une etoile a partir de la distance au centre du galaxie et son rayon

```
etoile etoile_crear(bint distanceCentreGalaxie, bint rayon);
```

// creer une etoile a partir de la distance au centre du galaxie, son rayon et volume

```
etoile etoile_crear(bint distanceCentreGalaxie, bint rayon, bint volume);
```

// creer une etoile de distance au centre du galaxie et rayon aleatoires, et calculer le volume a partir du rayon

```
etoile etoile_crearRnd();
```

***// obtenir etoile sur le clavier (demander distance au centre du galaxie et rayon.
Calculer volume a partir du rayon)***

```
etoile etoile_demander();
```

// information generale de l'etoile: DEBUG

```
void etoile_info(etoile &e, string teteCout = "");
```

2.3 Fichiers *galaxie.h* et *galaxie.cpp*

Afin de mieux illustrer l'utilisation du type de données "bint", une collection d'étoiles appelée

“galaxie” a été ajoutée :

// Collection d'etoiles

```
typedef struct nodeEtoile* collEtoiles;
```

// Node de la collection comportant une etoile et le pointeur au prochain node

```
typedef struct nodeEtoile {
```

```
    // l'etoile
```

```
    etoile etoile;
```

```
    // pointeur a l'etoile prochaine (et aux autres etoiles moins proches)
```

```
    collEtoiles suivNodeEtoile;
```

```
};
```

En tant que fonctions de base pour la création et le traitement d'une galaxie (collection d'etoiles), l'on compte dans le fichier **galaxie.cpp** avec l'implementation des fonctions suivantes:

// Cree collEtoiles vide

```
collEtoiles collEtoiles_creerVide();
```

// Cree pointeur au nouveau node de la collection d'etoiles

```
nodeEtoile* nodeEtoile_creer(etoile &e);
```

// Affiche l'information de la collection d'etoiles

```
void collEtoiles_afficherInfo(collEtoiles &c);
```

// Affiche toutes les etoiles de la collection

```
void collEtoiles_afficherToutes(collEtoiles &c);
```

// Affiche le nombre d'etoiles dans la collection

```
void collEtoiles_afficherQuantEtoiles(collEtoiles &c);
```

// Compte le nombre d'etoiles en la collection

```
unsigned int collEtoiles_quantiteEtoiles(collEtoiles &c);
```

// Ajoute une etoile a la collection

```
void collEtoiles_ajouterEtoile(collEtoiles &c, etoile e);
```

// Ajoute en tete une etoile a la collection

```
void collEtoiles_ajouteEnTete(collEtoiles &c, etoile e);
```

// Obtient la derniere etoile de la collection

```
etoile collEtoiles_derniereEtoile(collEtoiles &c);
```

// Obtient la plus grande etoile de la collection

```
etoile collEtoiles_plusGrandEtoile(collEtoiles &c);
```

// Calcule le volume total de la galaxie

```
bint collEtoiles_volumeTotal(collEtoiles &c);
```

// Obtient une etoile a partir de son ID

```
etoile collEtoiles_getEtoile(collEtoiles &c, unsigned long long id);
```

// Supprime une etoile de la collection

```
void collEtoiles_supprimerEtoile(collEtoiles &c, unsigned long long id);
```

2.4 Sous-programmes à détailler:

2.4.1. bint.cpp

```
void bint_ajouter(bint &a, unsigned long long b, int pos){
```

```
    // si l'array interne est tres petit
```

```
    if(pos>a.internal_nb-1) bint_setLongInter(a, pos+1);
```

```
    unsigned long long antVal = a.b[pos];
```

```
    a.b[pos]+=b;
```

```
    // si la pos predec pas utilisee
```

```
    if(pos > ((signed long)a.nb)-1) a.nb=pos+1;
```

```
    // si surcharge/overload
```

```
    if(a.b[pos]<antVal) bint_ajouter(a, 1, pos+1);
```

```
}
```

La fonction `bint_ajouter(bint &a, unsigned long long b, int pos)` est utilisée pour ajouter la valeur *b* de type *uint64* (*unsigned long long*) à la variable *a* de type *bint* en la position *pos* de son tableau dynamique interne.

Si l'array interne est très petit, on appelle `bint_setLongInter()` pour incrémenter la dimension de cette array interne. C'est à dire, créer un nouveau tableau de dimension plus grande que le dernière tableau en ajoutant les mêmes valeurs, et après détruire l'ancien tableau.

Après d'avoir obtenu un tableau interne assez grande, on ajoute la valeur *b* en position *pos*. Si on a ajouté cette valeur à une position qui n'était pas utilisée avant d'appeler `bint_ajouter()`, on a besoin d'incrémenter l'indicateur de blocs utilisées pour le *bint*. Si, après d'avoir ajouter la valeur *b* à la position *pos*, la valeur est devenu surchargée, on ajoute *1* a la prochain position de l'array interne.

2.4.2. *galaxie.cpp*

```
void collEtoiles_ajouterEtoile(collEtoiles &c, etoile e) {

    nodeEtoile* nouveauNodeEtoile = nodeEtoile_creer(e);
    if(c==null) c=nouveauNodeEtoile;

    else {

        if(bint_plusGrandQue(c->etoile.distCentGal,
nouveauNodeEtoile->etoile.distCentGal)) {

            // le nouveau est plus proche que le node en tete

            collEtoiles_ajouteEnTete(c, e);

        } else {

            // le node en tete es plus proche que le nouveau

            collEtoiles plusProcheNode = c;

            while(plusProcheNode->suivNodeEtoile!=null &&
bint_plusGrandQue(nouveauNodeEtoile-
>etoile.distCentGal, plusProcheNode->suivNodeEtoile-
>etoile.distCentGal)){

                plusProcheNode=plusProcheNode->suivNodeEtoile;

            }

            nouveauNodeEtoile->suivNodeEtoile=plusProcheNode-
>suivNodeEtoile;

            plusProcheNode->suivNodeEtoile=nouveauNodeEtoile;

        }

    }

}
```

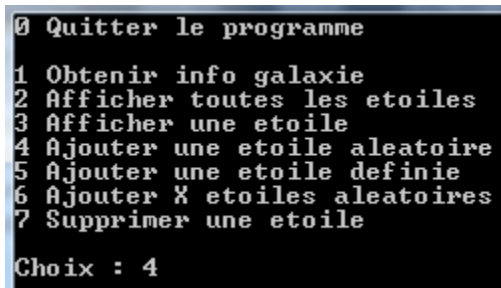
La fonction `collEtoiles_ajouterEtoile(collEtoiles &c, etoile e)` est utilisée pour ajouter une étoile `e` à la collection d'étoiles `c`. Ces étoiles sont triées par la distance au centre de la galaxie, donc on a besoin d'ajouter cette étoile entre l'étoiles qui se trouve immédiatement plus proche et celle qui se trouve immédiatement moins proche au centre de la galaxie que l'étoile `e` en question.

D'abord, cette fonction crée un nouveau enregistrement de type *nodeEtoile* avec l'étoile à

ajouter. Si la collection d'étoiles (galaxie) est vide, on assigne *e* à *c*, le pointeur au premier étoile. Si la collection n'est pas vide, mais la première étoile est moins proche au centre de la galaxie que l'étoile *e*, on appelle `collEtoiles_ajouteEnTete()` pour ajouter *e* en tête de la collection *c*. Si la première étoile est plus proche que l'étoile *e*, on cherche la dernière étoile plus proche au centre de la galaxie, et on ajoute la nouvelle étoile, entre cette étoile plus proche et la suivante.

3. Exemple d'utilisation

3.1 Affichage du menu principal



```
0 Quitter le programme
1 Obtenir info galaxie
2 Afficher toutes les etoiles
3 Afficher une etoile
4 Ajouter une etoile aleatoire
5 Ajouter une etoile definie
6 Ajouter X etoiles aleatoires
7 Supprimer une etoile

Choix : 4
```

Le **menu 1** sert à afficher l'information de la galaxie avec laquelle on va travailler. Cette information consiste en:

- La quantité d'étoiles dans la galaxie.
- La taille de la galaxie (la distance du centre de la galaxie à l'étoile la plus éloignée)
- L'information sur l'étoile la plus grand de la galaxie.
- Le volume total de la galaxie (la somme totale de tous les volumes des étoiles dans la galaxie)

Au début de l'exécution du programme, il n'y a pas d'étoiles.

Le **menu 2** sert à afficher les informations des étoiles dans la galaxie, automatiquement triées en ordre croissant basé sur la distance au centre de la galaxie.

Le **menu 3** sert à afficher les informations d'une étoile de nombre donnée.

Le **menu 4** sert à ajouter et afficher une étoile de distance et rayon aléatoire et calculer son volume.

Le **menu 5** sert à ajouter une étoile de distance et rayon précisés par l'utilisateur.

Le **menu 6** sert à ajouter et afficher un nombre d'étoiles définies par des distances et rayons aléatoires.

Le **menu 7** sert à supprimer une étoile identifiée avec un nombre donnée.

3.2 Exécution du programme

Lorsqu'on exécute le programme, il affiche le menu principal, où l'on peut choisir l'opération souhaitée.

- Utilisation de menu principal
- Affichage d'informations par défaut au commencement du programme.

```
0 Quitter le programme
1 Obtenir info galaxie
2 Afficher toutes les etoiles
3 Afficher une etoile
4 Ajouter une etoile aleatoire
5 Ajouter une etoile definie
6 Ajouter X etoiles aleatoires
7 Supprimer une etoile

Choix : 4
```

Tout d'abord, on choisit l'option *1* et confirme en appuyant sur la touche *enter* afin d'afficher l'information de la galaxie.

N'existant pas d'étoiles au commencement, le programme ne va afficher que le message *Quantite d'etoiles = 0* et rien d'autre.

Dans le cas où une ou plusieurs étoiles existent, le programme va afficher la taille de la galaxie, les étoiles qui y appartiennent, et l'information de l'étoile la plus grande et la plus éloignée du centre de la galaxie.

```
INFORMATION DU GALAXIE
Quantite d'etoiles : 5
Etoile plus eloignee (plus grand distance au centre) : 4294858300000000000000 km
Etoile plus grand:
  Distance au centre du galaxie : 17521069 x 10^12 km
  Rayon : <tres grand>
  Volume : entre 2^64 et (2^65)-1 x 10^9 km^3
Volume total du galaxie : entre 2^65 et (2^66)-1 x 10^3 km^3
```

Pour rentrer au menu principal, on appui sur n'importe quelle touche.

Dans le cas où il n'y a pas d'étoiles dans la galaxie, le programme ne va rien afficher si l'on choisit l'option 2. Dans un autre cas, il affiche les informations de toutes les étoiles.

```
Etoile #0 :
  Distance au centre du galaxie : 17521069 x 10^12 km
  Rayon : <tres grand>
  Volume : entre 2^64 et (2^65)-1 x 10^9 km^3
Etoile #1 :
  Distance au centre du galaxie : 20183629 x 10^12 km
  Rayon : 624 x 10^3 km
  Volume : 971882496 x 10^9 km^3
Etoile #2 :
  Distance au centre du galaxie : 37332774 x 10^12 km
  Rayon : <tres grand>
  Volume : entre 2^64 et (2^65)-1 x 10^9 km^3
Etoile #3 :
  Distance au centre du galaxie : 40438937 x 10^12 km
  Rayon : 19733 x 10^3 km
  Volume : 30735433223348 x 10^9 km^3
Etoile #4 :
  Distance au centre du galaxie : 42948583 x 10^12 km
  Rayon : 17823 x 10^3 km
  Volume : 22646568883068 x 10^9 km^3
```

On revient au menu principal et appui "3" pour afficher l'information d'une étoile désirée. Dans le menu suivant, on peut choisir le numéro d'identité d'une étoile, par exemple "4" ou "100".

```
Etoile #3
Distance au centre du galaxie : 40438937 x 10^12 km
Rayon : 19733 x 10^3 km
Volume : 30735433223348 x 10^9 km^3
```

Pour ajouter une étoile aléatoire à la galaxie, on choisit l'option "4". A chaque fois qu'on ajoute une étoile, elle va être triée dans un ordre croissant dans la galaxie, basée sur la distance au centre de la galaxie.

```
Etoile cree :
Distance au centre du galaxie : 4579232 x 10^12 km
Rayon : 1990 x 10^3 km
Volume : 31522396000 x 10^9 km^3
Quantite d'etoiles : 7
```

Si on a besoin d'ajouter une nouvelle étoile avec des paramètres choisis par l'utilisateur, on choisit l'option 5.

Le programme va ensuite demander la distance de l'étoile au centre de la galaxie et son rayon. Par exemple on peut mettre "500000" pour la distance et "2000" pour le rayon. Tout comme le cas précédent, la nouvelle étoile va être mise automatiquement dans l'ordre approprié dans la galaxie.

```
Creer nouveau etoile :
Distance au centre du galaxie en 10^12 km (ex Soleil est 250000) : 3456789
Rayon en 10^3 km (ex. Soleil est 1392) : 1234
Volume calculee : 7516323616 x 10^3 km^3
Nouveau etoile ajoutee
Quantite d'etoiles : 8
```

Afin d'ajouter une ou plus d'étoiles aléatoires, on choisit l'option "6". Le programme va alors attendre à ce qu'on précise le nombre d'étoiles à ajouter. Par exemple, on peut mettre "100" et attendre jusqu'à ce que toutes les étoiles soient créées. Après avoir en fini, on peut appuyer sur n'importe quelle touche pour rentrer au menu principal.

```
Quantite d'etoiles a ajouter : 5
Attente pour la creation d'etoile #1...
Attente pour la creation d'etoile #2...
Attente pour la creation d'etoile #3...
Attente pour la creation d'etoile #4...
Attente pour la creation d'etoile #5...
Quantite d'etoiles : 5
```

Le dernière menu (menu 7) sert à supprimer une étoile en tapant son nombre (ID). Si l'étoile avec le nombre saisi existe, elle sera éliminée, en conservant l'ordre des étoiles dans la galaxie.

```
Quantite d'étoiles : 5  
En train d'éliminer l'étoile #4...  
Quantite d'étoiles : 4
```

Dans le menu principal, on peut refaire toutes les étapes précédentes. Pour terminer le programme, on appui sur les touches "0" ou "X".

4. Conclusion

Ce projet sert bien comme un exemple d'implémentation des structures de données telles que enregistrements et listes chaînées pour applications demandant l'utilisation des nombres entiers très grandes, dont la capacité de calcul des ordinateurs actuels ne permet pas son traitement en tant que types de données simples.

En outre, le travail d'implémentation de tels outils nous permettre aussi d'avoir une compréhension plus approfondie sur l'utilisation des bibliothèques des nombres de précision arbitraire telles que [C++ Big Integer Library](#), [apfloat](#) parmi d'autres.